# Position Statement for New Paradigms for Internetwork Security Panel

Steven J. Greenwald
Email: `greenwald@itd.nrl.navy.mil`
WWW: `http://www.itd.nrl.navy.mil/ITD/5540`

*Center for High Assurance Computer Systems*
*Naval Research Laboratory*
*Washington, DC 20375*
*United States of America*

## Introduction

The security policy currently used on most distributed systems is an old one, dating back to simpler times when most computer systems were centralized. This security policy is based on the idea that there is a central managing authority, called the *system administration*, that is ultimately responsible for the management of computer security within an administrative domain. In this security policy system administration includes the management of system resources, user accounts, and user privileges. This security policy is typified by an operating system such as UNIX. I refer to this older security policy as the *Jurassic Age Security Policy* (JASP) since it apparently dates back to the time when huge dinosaur computers were kept in air-conditioned pens, lazily grazing on their data, before faster, leaner machines wiped them out.[1]

[1] I am obviously open to suggestions for more appropriate terminology, and I'm also interested in exactly when JASP first came into existence.

JASP introduces difficulties when working in a distributed computing environment, and most of the computer systems in use on the Internet are based on JASP. I am specifically concerned with the management of system resources and access control in a distributed computing environment. We need a new paradigm for security that is congruent with the highly distributed nature of the Internet.

## Paradigm Problems

JASP presents the following problems when working in a distributed environment.

1. User-names are often duplicated across name-space domains in a distributed system. For example, two different users may have the same user-name on two different hosts within a distributed system.

2. Location transparency may not be possible. For mobile users who often change hosts, the combination of user-name and host-identifier fails to uniquely identify the user. One user may have two (or more) different user-names at different locations. Two users in different administrative domains may have the same user-name.

3. There exists a "weak link in the chain" effect. The security of the entire distributed system depends upon the security of the individual hosts that are being used within a group of administrative domains. One lax system administration can compromise an entire distributed system.

4. Users often need to assume different roles, and JASP does not accommodate this. I define a role as a labeled set of capabilities that a user can activate. Roles, as opposed to protection groups, are generally considered to be a form of mandatory access control. For example, a user may wish to simultaneously assume the roles of "panelist" and "chair" for a particular session.

5. It is often very difficult to share resources with other users on other computer systems without getting permission from the system administrations involved. Especially for real-time applications.

6. Foreign user accounts are often necessary to correct the previous problem. This places a management burden on the system administration and there is the very serious difficulty of the system administration initially verifying the identity of these foreign users. In addition, foreign user accounts present the potential problem of giving the foreign user too many permissions.

7. Military chain of command systems and corporate hierarchical systems may be difficult to model and implement because their structure clashes with the "flat" structure of the omnipotent-system-administrator approach of JASP.

## Solution Requirements?

There are many ways to solve the above stated problem. I believe the best solution will contain elements of a libertarian (classical liberal) philosophy that maximizes the freedom of users while limiting system administration intervention to only vitally necessary functions. Philosophically, this should have the benefits of allowing users as much flexibility in managing their affairs as possible, while eliminating much of the drudgery commonly associated with system administration. I believe a this approach is a good compromise between authoritarian control and anarchy. I believe this because of the common observation that the Internet is the closest thing to a workable, successful anarchy that the modern world has ever developed. Yet it is this very anarchy that is now causing our present security concerns.

In a libertarian approach, users would be give more power than they currently have. This does not mean that system administrators need give up any of their power or control. In fact, it will probably mean that system administrators will be giving up the things they commonly associate with drudgery.

Since user processes and resources are for all intents now decentralized in many distributed systems, it makes sense to decentralize the method of access control, and the method of resource management.

First, we can do away with the requirement that applications identify users by operating system dependent user names and paths. I believe that role based access control (RBAC) is the preferred way to solve this problem. At a minimum, a role would need to be composed of a label (name), a set of capabilities, and a list of users that are members of that role. In addition, roles can be designed to be related to users in a many-to-many way, so that users can effectively share the same role (many users to one role) and individual users can be members of more than one role (many roles for one user). If required, auditing of users can still take place, even at the operating system level.

With RBAC, we gain several advantages. The name of the role can be more descriptive than often cryptic user names, anonymity is possible, the many-to-many relationship allows users to assume different roles, and more than one user to use the same role. The management of roles becomes part of the particular distributed application, instead of an operating system dependent issue. With RBAC, system administrators would not be pestered with user requests for foreign accounts, requests to add users to protection groups, and so forth, since these functions can be handled by users activating other roles.

Resource management is the other area where our solution lies. Currently, all resources are, in some sense, "owned" by the administrative domain they belong to. This is the wrong paradigm to use in today's decentralized world. Looking at this from a libertarian point of view, it would be better if users could logically "own" the resources they have been allocated, and deal with them as they see fit (in a secure way, of course), allowing for things such as n-person rules, different decision support mechanisms, and so forth.

For example, if a user has a certain amount of storage space allocated, why can't that user let other users access that storage space, without having to pester a system administrator? This is a common problem in systems such as UNIX, where only someone with the highest permissions can add someone to a protection group. It makes more sense to allow individual users to perform these functions, since they have already made the decision.

In addition, there should be no reason to logically view these resources as belonging to particular centralized machines. Users should be allowed to logically share their resources across administrative domain boundaries, and use them as they see fit (*e.g.,* in collaborative ways such as multiple authors writing a paper in real-time).

Utilities and security policies can and should be designed to accommodate these necessary elements. Some of the issues to be solved in these policies are things such as the exact mechanism of RBAC, how to manage resources efficiently across administrative domains, how to handle the name-space that will occur with such systems, and how to organize the combination of RBAC and distributed resource management in a coherent manner that users can understand and use.

But the most important goal of all is that we must free users from a large amount of dependence on various administrative domains, while simultaneously freeing the various system administrations from many tedious tasks. I believe that this point will become increasingly important as distributed systems continue to multiply.